3-1-2012

A Framework for Evaluating Traceability Benchmark Metrics

Yonghee Shin
DePaul University, Chicago, IL, U.S.A., yshin@cdm.depaul.edu

Jane Huffman Hayes
University of Kentucky, Lexington, KY, U.S.A., hayes@cs.uky.edu

Jane Cleland-Huang
jhuang@cs.depaul.edu

A Framework for Evaluating Traceability Benchmark Metrics

Yonghee Shin[1,2], Jane Huffman Hayes[1,3], and Jane Cleland-Huang[1,2]

[1] Center of Excellence for Software Traceability (CoEST)
http://www.coest.org/
[2] DePaul University, Chicago, IL, U.S.A.
`yshin@cdm.depaul.edu, jhuang@cs.depaul.edu`
[3] University of Kentucky, Lexington, KY, U.S.A.
`hayes@cs.uky.edu`

Abstract. Many software traceability techniques have been developed in the past decade, but suffer from inaccuracy. To address this shortcoming, the software traceability research community seeks to employ benchmarking. Benchmarking will help the community agree on whether improvements to traceability techniques have addressed the challenges faced by the research community. A plethora of evaluation methods have been applied, with no consensus on what should be part of a community benchmark. The goals of this paper are: to identify recurring problems in evaluation of traceability techniques, to identify essential properties that evaluation methods should possess to overcome the identified problems, and to provide guidelines for benchmarking software traceability techniques. We illustrate the properties and guidelines using empirical evaluation of three software traceability techniques on nine data sets. The proposed benchmarking framework can be broadly applied to domains beyond traceability research.

1 Introduction

Sim et al. [32] challenged the software engineering community to perform benchmarking as a means to advance research in addition to identifying state-of-the-art techniques. In fact, the benefits of benchmarking for any scientific community are manifold: reaching consensus on the community's research goals, fostering communication and collaboration among research groups, executing rigorous examination of existing techniques leading to further improvement [32], and enabling practitioners to select the right techniques. In response to this challenge, the software traceability research community is actively engaged in developing a framework to support the benchmarking of traceability techniques [10, 11, 21].

Software traceability is broadly recognized as a critical activity in the development of many non-trivial software systems. Defined as the ability to establish connections between software engineering artifacts such as requirements, design, code, and test cases [17, 29], traceability is essential for ensuring that the developed product conforms to its requirements, that all requirements are covered in the design and code, and for supporting maintenance activities such as impact analysis and regression test selection. Unfortunately, the task of creating and maintaining traceability links is highly labor-intensive and time consuming, and is often perceived by organizations as being prohibitively expensive [7]. For this reason, various techniques have been proposed to automate the tracing process [4, 9, 18, 27, 28] and to use human feedback to improve the accuracy of generated traces [12, 24].

An important first step in the process of benchmarking for traceability techniques is to define evaluation methods including evaluation metrics. Recent published studies in software traceability have used a broad range of evaluation metrics suggesting that there is little community consensus on which metrics to use, or how those metrics should be computed. This discrepancy in the use of metrics was confirmed by a survey at the 6th International Workshop on Traceability in Emerging Forms of Software Engineering(TEFSE'11). Survey respondents were asked to review a list of possible metrics and metric computation methods and to select the ones that they felt should be used by the community to support various kinds of evaluation. Results from this survey showed limited agreement between the respondents. While each research project has its own goals and objectives, and therefore may need to customize the selection of evaluation methods, there is still a clear need to define a standard set of evaluation methods in order to make fair comparisons across traceability techniques. Furthermore, standardized evaluation methods are useful for meta-analyses [22] which enable the community to use results from published studies to accumulate higher level knowledge.

Given the lack of consensus in the traceability research community where evaluation methods are concerned, we need to ask and answer the meta-question of 'what criteria should be used to select the best evaluation methods for benchmarking?' This paper addresses this question by investigating current practices in evaluating software traceability techniques, proposing a framework that consists of a set of desirable properties that can be used to evaluate evaluation methods for benchmarking purposes, and providing guidelines for evaluating software traceability techniques.

To investigate the current practices for evaluating software traceability techniques, we performed a systematic literature review [22, 23], from which we observed three dimensions of diversity in evaluation methods and identified six desired properties of evaluation methods. In this paper we explain each property with examples taken from an empirical evaluation of three traceability techniques across nine different data sets.

The remainder of the paper is organized as follows: Section 2 provides background on traceability research and discusses related work. Section 3 describes results from the systematic literature review. Section 4 explains the traceability techniques and data sets used for the case study. Section 5 details the evaluation framework and provides guidelines for evaluation. Section 6 addresses threats to validity of our study. Finally, Section 7 summarizes our study and discusses future work.

2 Background and Related Work

We present the basic concepts and terms in software traceability and discuss prior work related to our study.

2.1 Software Traceability Techniques

The Center of Excellence for Software Traceability (CoEST) [2] defines a *trace link* as a 'specified association between a pair of artifacts, one comprising the *source* artifact and one comprising the *target* artifact.' The task of tracing therefore involves discovering the set of target artifacts that are related to a given source artifact, and then establishing trace links between them. This can be accomplished manually or in a semi-automated way. For example, Information Retrieval (IR) techniques, such as the Vector Space Model (VSM) [26] can be used to automatically compute the degree of relevance between source and target artifacts. The results are typically sorted in descending order according to relevance score, and pairs of artifacts scoring over a certain threshold are classified as *candidate links* and are said to be *retrieved*. Following normal IR convention, a source artifact is often referred to as a *query* and a target artifact as a *document*.

To evaluate the accuracy of a traceability technique, the retrieved results are compared against a predefined *answer set*. The accuracy of traceability techniques is typically measured either using *classification accuracy metrics* or *rank accuracy metrics*. Classification accuracy metrics [19] count the number of correctly or incorrectly retrieved links. Rank accuracy metrics [19] measure the accuracy of the relative ordering of correct links in the ordered retrieval results. Some studies, such as those studying the behavior of human trace analysts [12], do not use relevance scores, hence use only classification accuracy metrics. A precise definition of each metric will be provided in Section 3.

2.2 Related Work

As evaluation is a critical component in any science and technology field [30], the study of evaluation methods has received ongoing attention in many computer science research areas including software engineering. Some studies have examined classification accuracy and rank accuracy metrics in focused research areas [19, 25, 33, 35], others have provided guidelines for using statistical testing to compare techniques [5, 14].

Sundaram et al. [33] evaluated traceability techniques using classification accuracy metrics and rank accuracy metrics and emphasized the role of the latter. Herlocker et al. [19] performed an extensive study on the evaluation of collaborative filtering recommender systems. Our correlation analysis between metrics was inspired by their work. Weyuker et al. [35] and Ma et al. [25] also examined multiple classification and rank accuracy metrics for software fault prediction models.

Arcuri et al. [5] provided guidelines for statistical testing to evaluate randomized algorithms that are used in software engineering. Demšar [14] suggested statistical comparison methods for machine learning classifiers focusing on evaluation with multiple data sets. While our study recommends statistical testing, our focus is not on suggesting specific statistical testing methods, but on providing a general framework to help assist with evaluation methods.

A comprehensive set of issues in evaluation of information retrieval techniques was discussed by Saracevic [30]. While he rightly addresses relevant issues in evaluation, the study was focused on discussing problems, not on providing practical guidance for evaluation in general, much less for benchmarking with a specific task.

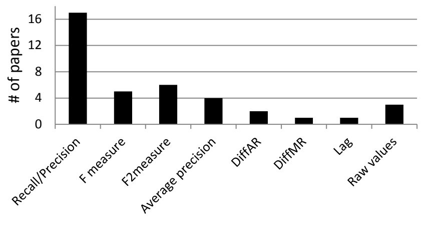
In general, prior studies have focused on parts of evaluation without providing a unified view and/or lack of practical guidance for benchmarking. The properties defined in our study are designed specifically for benchmarking purposes, and can be broadly applied beyond the specific evaluation methods and traceability techniques used in our case study.

Although not related to *evaluation metrics*, two studies on evaluation of *software metrics* performed by Weyuker [34] and Briand et al. [8] are worth mentioning. Software metrics, such as using lines of source code as an indicator of code complexity, are frequently used to estimate software quality or development effort. These two studies defined mathematical properties that software metrics should satisfy. A simple example is non-negativity [8]. These properties have been widely used to evaluate software metrics and to help develop new metrics. Although we do not use a mathematical formulation to define the properties of evaluation metrics, the goal of our study and theirs are coincident in that their studies and ours aim at providing a *framework* to guide selection or design of appropriate metrics.

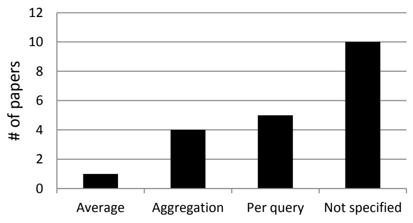
The novel contribution of our work in comparison to these earlier studies is that it provides a framework for selecting evaluation methods for benchmarking and works through its properties using an empirical evaluation of traceability techniques.

3 Current Practice

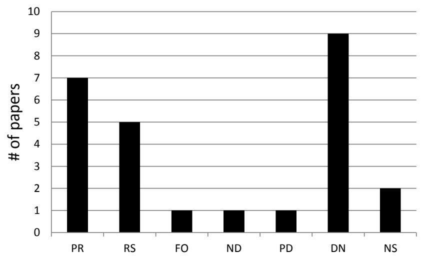
To investigate current practices for evaluating software traceability techniques, we performed a systematic literature review [23]. For this purpose, we selected papers that had been recently published in the following ten relevant journals, conferences, and a workshop in 2010 and 2011: IEEE Transactions on Software Engineering (TSE), ACM Transactions on Software Engineering and Methodology (TOSEM), Empirical Software Engineering Journal (ESE), Requirements Engineering Journal (REJ), International Requirements Engineering Conference (RE), International Conference on Software Engineering (ICSE), International Conference on Software Maintenance (ICSM), Foundations of Software Engineering (FSE), Automated Software Engineering (ASE), and finally the Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE). We used the search term *trac* to retrieve papers containing terms such as *traceability, trace*, and *tracing* in the title. Among the retrieved papers, we included only full length



(a) Metrics



(b) Summarization Methods



(c) Threshold Types

Fig. 1. Current practice of metrics usage

regular research papers and excluded papers that did not include quantitative evaluation. As a result we found 24 relevant papers.

We limited the scope of our literature review to the *accuracy* evaluation of traceability techniques, excluding other factors such as the estimation of human effort. We analyzed evaluation results reported in any format including graphs and tables. As a result of this survey, we identified seven metrics that have been used to evaluate software traceability techniques. We also identified several variants of these metrics. These variants primarily reflect different ways of classifying relevant versus irrelevant links and different ways of summarizing results. These metrics and their variants are described in the following section.

3.1 Metrics

Figure 1(a) shows the frequency of usage for each of the identified metrics. *Recall*, *precision*, and *F_β-measure* are classification accuracy metrics, while *Average Precision (AP)*, *DiffAR*, *DiffMR*, and *Lag* are rank accuracy metrics. Some studies reported *raw values* such as a count of correct or incorrect links or relevance scores without any further computation or aggregation. Overall, recall and precision were the most frequently used metrics, followed by F_β-measure. Note that the number of studies in each figure in Figure 1 is over 24 because some studies used multiple metrics.

Classification Accuracy Metrics *Recall* [26] measures the fraction of relevant documents that are correctly retrieved and is defined as follows:

$$Recall = \frac{|CorrectlyRetrievedDocuments|}{|RelevantDocuments|}. \tag{1}$$

Precision [26] measures the fraction of retrieved documents that are relevant and is defined as follows:

$$Precision = \frac{|CorrectlyRetrievedDocuments|}{|RetrievedDocuments|}. \tag{2}$$

F_β-measure [26] is a weighted harmonic mean of recall and precision and is defined as follows:

$$F_\beta\text{-}measure = \frac{(1 + \beta^2) \times precision \times recall}{(\beta^2 \times precision) + recall}, \tag{3}$$

where $\beta > 1$ puts more emphasis on the importance of recall. F-measure is a simplified expression of F_1-measure.

Rank Accuracy Metrics *Average precision* [26] measures the extent to which a traceability technique places correctly retrieved links towards the top of the ranked list and is defined as follows:

$$AveragePrecision = \frac{\sum_{r=1}^{N}(P(r) * isRelevant(r))}{|RelevantDocuments|}. \tag{4}$$

where r is the rank of a document in the ordered list of retrieved results from N documents, $isRelevant()$ is a binary function assigned 1 if the rank is relevant and 0 otherwise, and $P(r)$ is the precision computed after truncating the list immediately below that ranked position.

DiffAR [33] measures the difference between the average relevance scores of correctly and incorrectly retrieved links and is defined as follows:

$$DiffAR = \frac{\sum_{(q,d)\in L_T} rel(q,d)}{|L_T|} - \frac{\sum_{(q,d)\in L_F} rel(q,d)}{|L_F|}, \tag{5}$$

where q is a query, d is a document, $rel(q, d)$ is a relevance score between q and d, L_T is the set of correctly retrieved links, and L_F is the set of incorrectly retrieved links.

DiffMR [33] measures the difference between the median relevance scores of correctly and incorrectly retrieved links and is defined as follows:

$$DiffMR = med_{(q,d)\in L_T}(rel(q,d)) - med_{(q,d)\in L_F}(rel(q,d)). \tag{6}$$

However, as Sundaram et al. mentioned [33], metrics such as DiffAR and DiffMR, which use relevance scores to measure the internal structure of a ranked list, are unstable compared with metrics that use the relative position of links in the ranked list. This is because each technique can produce a very different range of relevance scores. For example, one technique may have a maximum relevance score of 0.2, while another might have a maximum score over 0.9. Due to this instability we do not consider these two metrics further in this study.

Lag [33] measures how many incorrect links are retrieved above each correct link and is defined as follows:

$$Lag = \frac{\sum_{(q,d)\in L_T} Lag(q,d)}{|L_T|}, \tag{7}$$

where L_T is the set of correctly retrieved links and $Lag(q,d)$ is the number of incorrectly retrieved links with higher relevance scores than a correctly retrieved link (q,d). A low Lag value indicates that the technique traced accurately.

In addition to the metrics that have been used in recent traceability research, we additionally examine *Area Under the ROC curve (AUC)* [15] because it has been frequently used as a classification accuracy metric and has potential applicability for evaluating software traceability techniques in the future. *AUC* is the area under a Receiver Operating Characteristic (ROC) curve, where each point in a ROC curve is a pair of values representing recall and False Positive (FP) rate, respectively. An FP rate is computed as the fraction of non-relevant documents that are incorrectly retrieved. AUC measures how well a traceability technique discriminates between relevant and non-relevant documents.

3.2 Result Summarization Methods

One unique aspect of traceability, compared to many other information retrieval tasks and classification problems, is that a single data set has multiple associated queries. Therefore, the tracing results from these multiple queries must be summarized to provide an overall evaluation of a software traceability technique. Our literature survey revealed three common approaches for summarizing such results. These are (1) averaging metric values from the results of each individual trace query (*average method*), (2) computing a metric value after ordering the retrieved documents in descending order of relevance scores across queries (*aggregation method*), and (3) reporting per-query tracing results using statistical tests or graphs (*per-query method*).

Figure 1(b) shows the distribution of usage frequency for the summarization methods. The use of the per-query method was obvious from the descriptions and graphs presented in papers. However, in many cases the reported experiments did not clearly describe their summary metric computations, and it was therefore unclear whether they used the average or the aggregation method. Of the documented methods, many more studies used the aggregation method than the average method, we therefore conjecture that a large number of studies in the *not-specified* category implicitly used the aggregation method. Some studies used multiple summarization methods by using the average method to compute one metric and the aggregation method to compute others. This result indicates that the traceability research community requires better guidance on the use of metrics, and furthermore, that they need to reach consensus in order to fairly evaluate the efficacy of various techniques.

3.3 Classification Threshold Types

Our survey results showed that five types of thresholds were used in conjunction with classification accuracy metrics. These included (1) a recall level at which precision was measured (PR), a relevance score (RS), an F-measure-optimizing point (FO), the number of retrieved documents (ND), and the percentage of retrieved documents (PD). The techniques that do not compute relevance scores (e.g., manual tracing) did not need to use thresholds (DN in Figure 1(c)). A few studies did not specify the threshold type (NS). The most commonly adopted approach for techniques that required thresholds was to measure precision at fixed recall levels.

3.4 Summary of Observations

Based on results from this literature review, as well as our own experience of engaging in traceability research projects, we identified the following challenges for adapting evaluation methods for benchmarking purposes.

– *Observation 1.* Evaluation metrics are sometimes used which are not appropriate for evaluating the task being performed.
– *Observation 2.* Some evaluation methods are not feasible for benchmarking, although the evaluation methods can be used for individual studies.
– *Observation 3.* Too many variations of evaluation methods have been used without in-depth investigation into their differences.
– *Observation 4.* Comparison results change depending on measurement variants such as thresholds.
– *Observation 5.* Statistical testing has rarely been used, leaving the threat to conclusion validity unaddressed.
– *Observation 6.* No standard technique has been defined for summarizing evaluation results across multiple data sets.

Each of these observed problems implies a property that must be considered when selecting evaluation methods, and which should be satisfied for benchmarking purposes. For example, Observation 1 highlights the need to select evaluation metrics that are suited to the goals of the study (*goal satisfiability* property). Observation 2 leads to the need to select evaluation methods that facilitate benchmarking (*generalizability* property). From the observed problems and from our own experience, we identified six such properties that an evaluation method should possess in order to be suited for benchmarking. In Section 5, we explain each property in detail with examples drawn from our case study. The case study is explained next.

4 Case Study Setup

In this section, we present a case study that will be used throughout the remainder of the paper to explain the properties which evaluation methods must exhibit in order to be used for benchmarking. The case study includes both qualitative and quantitative analysis of trace results produced by three different tracing techniques. For purposes of quantitative analysis, we empirically evaluate the Vector Space Model (VSM) and the two variants: the VSM Global approach [13] and the Rocchio approach [6, 31]. These variants were chosen because they produce markedly different tracing results from the baseline VSM approach. The evaluation is conducted using nine different data sets. This section describes these tracing techniques and the data sets and then reports the tracing results using a subset of metrics previously defined in Section 3.

4.1 Vector Space Model

In VSM, each query q and each document d is represented as a vector of terms $T = t_1, t_2, ..., t_n$ which is the set of all terms in the set of queries. A document d is therefore represented as a vector $\boldsymbol{d} = (w_{1,d}, w_{2,d}, ..., w_{n,d})$, where $w_{i,d}$ represents the term weight of term i for document d. A query is similarly represented as $\boldsymbol{q} = (w_{1,q}, w_{2,q}, ..., w_{n,q})$. The standard weighting scheme known as tf-idf is used to assign weights to individual terms, where tf is the term frequency and idf is the inverse document frequency. Term frequency is computed for document d as $tf(t_i, d) = (freq(t_i, d))/(|d|)$, where $freq(t_i, d)$ is the frequency of the term in the document and $|d|$ is the length of the document. Inverse document frequency, idf, is typically computed as:

$$idf_{ti} = log_2(n/n_i), \tag{8}$$

where n is the total number of documents and n_i is the number of documents in which term t_i occurs. The individual term weight for term i in document d is then computed as $w_{id} = tf(t_i, d) \times idf_{ti}$. A similarity score, $sim(d, q)$, between document d and query q is computed as the cosine of the angle between the two vectors as follows:

$$sim(d, q) = \frac{(\sum_{i=1}^{n} w_{i,d} w_{i,q})}{\left(\sqrt{\sum_{i=1}^{n} w_{i,d}} \cdot \sqrt{\sum_{i=1}^{n} w_{i,q}}\right)}. \tag{9}$$

4.2 Global IDF Approach

The role of *idf* in equation 8 is to assign higher weights to rarer terms. The assumption here is that rarer terms within the project-specific artifacts provide more information for tracing than commonly-occurring terms. However, this weighting scheme can result in false positives when terms that are quite common in general language usage occur a disproportionately few times in the traced dataset. In this case, the terms may be assigned high weights by a local *idf* algorithm even though they are relatively unimportant in general use.

To address this problem, the *idf* can be computed against a more general corpus such as the American National Corpus (ANC) as follows:

$$idf\_ANC_{ti} = 1/min(rf_{ti}, 1000) \tag{10}$$

where rf_{ti} is the relative frequency assigned to term ti in the ANC collection. Furthermore, if term ti is not found in the ANC, the idf_ANC_{ti} is set to 1, as this suggest that the term is extremely rare.

We call the approach that uses a local data set to compute *idf the VSM Local IDF approach*, while the approach that uses the ANC is called *the VSM Global IDF approach*.

4.3 Rocchio Approach

In the Rocchio approach, relevance scores are iteratively computed based on user feedback. In each iteration, the top N links (with N typically set to 3-5) are presented to the user for relevance feedback. If a document is marked as relevant by the user, then the weight of any term that appears in both the document and the query is increased in the query's term vector. For the initial iteration, we used the VSM to compute the weights for the query and document vectors. Then, we simulated perfect user feedback according to whether a link was in the answer set or not. Five iterations of feedback were collected, and in each iteration the top five links were used to compute the new query vector.

4.4 Data Sets

This study included nine data sets with traces between various types of artifacts including requirements, use cases, interaction diagrams, class code, and test cases. Table 1 summarizes the data sets.

4.5 Preliminary Results

Table 2 and Figure 2 report the metric values and distributions of AP, AUC, Lag, and precision at a recall value of 1.0 ($\text{PR}_{1.0}$)[4] computed using the aggregated summarization method for the three techniques across the nine data sets.

[4] In this study, we used interpolated precision defined in [26], in the case that precision cannot be measured at fixed recall levels.

Table 1. Datasets used in the Case Study

No.	Dataset	Description	Source		Target		# of Links
			Name	No	Name	No	
1	**Albergate (AL)** [1,4,27]	Hotel management system	High level reqs.	17	Classes	55	54
2	**CM1-subset (CM)** [3]	Space telescope system	High level reqs.	22	Low level reqs.	53	45
3	**eAnci (EA)** [16]	Italian municipalities management system	Use cases	140	Classes	55	567
4	**Easy Clinic (ECT)** [3]	Electronic health care system	Classes	47	Test cases	63	204
5	**Easy Clinic (EUC)** [3]	Electronic health care system	Use cases	30	Classes	47	93
6	**E-Tour (ET)** [3]	Tour guide system	Use Cases	58	Classes	116	308
7	**Gantt (GA)** [20]	Project management tool (Gantt charts)	High level reqs.	17	Low level reqs.	69	68
8	**SMOS (SM)** [16]	High school student monitoring system	Use cases	67	Classes	100	1044
9	**WV-CCHIT (WV)** [31]	Health information system	Requirements	116	Regulatory code	1064	587

Table 2. Tracing Results using Aggregated Summarization Method

	AP			**Lag**			**AUC**			**PR$_{1.0}$**		
Dataset	**G^{a}**	**L^{b}**	**R^{c}**	**G**	**L**	**R**	**G**	**L**	**R**	**G**	**L**	**R**
Albergate	0.20	0.28	0.48	187.72	159.31	157.93	0.79	**0.82**	**0.82**	**0.06**	**0.06**	**0.06**
CM1-subset	0.12	0.42	0.50	253.67	102.73	76.91	0.77	0.91	0.93	**0.05**	**0.05**	**0.07**
eAnci	0.09	0.11	0.20	3264.72	3136.59	3872.47	0.54	0.56	0.46	**0.07**	**0.07**	**0.07**
Easy Clinic CC-TC	0.22	0.30	0.60	890.32	613.78	423.13	0.68	0.78	0.85	0.08	**0.09**	**0.09**
Easy Clinic UC-CC	0.17	0.61	0.85	340.70	95.63	44.19	0.74	0.93	0.97	0.08	**0.10**	**0.10**
E-Tour	0.06	0.28	0.33	2608.72	1515.65	1301.96	0.59	0.76	0.80	**0.05**	**0.05**	**0.05**
Gantt	0.22	0.35	0.40	242.38	123.76	98.28	0.78	0.89	0.91	0.08	**0.09**	**0.09**
SMOS	0.23	0.26	0.32	2072.66	2212.97	2028.03	0.63	0.61	0.64	0.16	**0.17**	**0.17**
WV-CCHIT	0.04	0.16	0.27	30906.66	6391.38	5631.80	0.75	**0.95**	**0.95**	**0.01**	**0.01**	**0.01**

[a] Global IDF [b] Local IDF [c] Rocchio

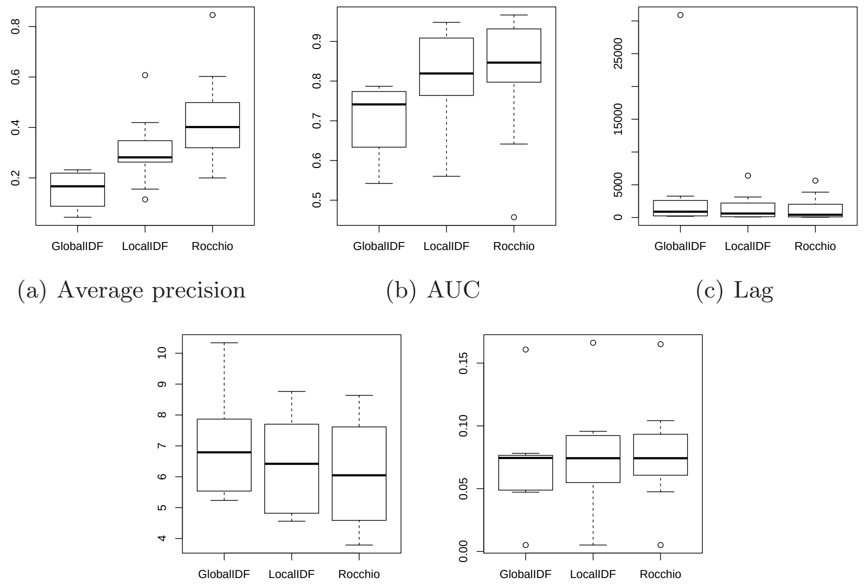
Because Lag from WV-CCHIT was an outlier, we also present the log scaled Lag in Figure 2(d) to visualize the trend more clearly. Note that for Lag, a high value indicates low trace accuracy. In Table 2, the techniques with tied metric values for a data set are presented in bold.

5 Desirable Properties

We now define each property, illustrate its importance with examples from the empirical case study, and provide guidelines for benchmarking.

5.1 Goal Satisfiability

An evaluation metric should be able to measure how well a traceability technique achieves the goal of tracing. *Goal satisfiability* is the ability for a metric to mea-



(a) Average precision (b) AUC (c) Lag

(d) Log of Lag (e) Prec. at Recall 1.0

Fig. 2. Tracing results using aggregated summarization method

sure the extent to which a technique achieves the given goal of the benchmarking task.

Examples The typical goals of trace techniques are:

- Goal 1: To find all the relevant documents with high accuracy. This is especially important for coverage analysis, where missed relevant documents can lead to redundant effort for reimplementation.
- Goal 2: To find relevant documents without inclusion of irrelevant documents. This is important to reduce unnecessary effort for the human analysts who vet the trace retrieval results.
- Goal 3: To accurately rank the most relevant documents near the top of the retrieved list. This further reduces human effort and increases a user's trust in the tracing tool, which is essential for adoption purposes.

A metric should evaluate the achievement of at least one of these goals. Precision alone fails to measure the achievement of Goal 1 by ignoring untraced relevant documents. Recall alone fails to measure the achievement of Goal 2 by ignoring incorrectly traced documents. F_β*-measure* fails to measure the achievement of both Goals 1 and 2 by obscuring recall and precision, although a very high F_β*-measure* or a very low F_β*-measure* indicates both recall and precision are very high or very low, respectively. Precision at certain recall levels measures the achievement of both Goals 1 and 2. All these metrics, however, fail to evaluate the achievement of Goal 3 by assigning the same weights to all correctly retrieved links. AP, Lag, and AUC evaluate the achievement of Goal 3. However, the three metrics use different weighting schemes to represent the position of a correct link in the ranked result list. AP assigns a non-proportionally higher weight to a correct link ranked at the top of the result list than to a correct link ranked at the bottom of the result list. On the other hand, Lag assigns a non-proportionally higher weight to a correct link ranked at the bottom of the result list. As a result, AP highly rewards techniques that rank correct links at the top of the result list while Lag severely penalizes the techniques that rank correct links at the bottom of the result list. AUC assigns proportionally higher weight to a correct link ranked at the top of the result list than to a correct link ranked at the bottom of the result list.

Guidelines Benchmarking should use evaluation metrics that are consistent with the goal under evaluation. When multiple candidate metrics are available that provide similar goal satisfiability, metric selection can be based on the satisfaction of other properties.

5.2 Generalizability

An evaluation method that suits a single study within a certain context of techniques and data sets may not be general enough for benchmarking, which uses more diverse sets of techniques and data sets. *Generalizability* is the ability for an evaluation method to provide coverage of diverse datasets and traceability techniques in order to support benchmarking.

Examples In software traceability, certain value-based classification thresholds such as the number of retrieved documents or a relevance score cannot be used when a selected threshold is out of range of the values available in benchmarked techniques and data sets. For example, the smallest number of target documents in the nine data sets is 47 in Table 1, creating the constraint that the 47th document is the maximum viable classification threshold if the number of retrieved documents is used as the threshold. As another example, Figure 3 shows the wide variability in relevance scores per query as a result of applying the Global IDF approach across the nine data sets. This variability makes it difficult to identify an appropriate relevance score threshold for classification purposes. Customizing thresholds according to different datasets and techniques could introduce unfairness issues or could bring the validity of the benchmark into question. Therefore, although the number of retrieved documents and a relevance score can be used for an analysis of retrieval results for a single study, it is not an appropriate choice for benchmarking across multiple datasets.

The DiffAR and DiffMR metrics defined in Section 3 also violate the generalizability property for similar reasons.

Guidelines For benchmarking purposes, we therefore recommend using relative values instead of absolute values. For example it is better to use the percentage of retrieved documents as opposed to the number of retrieved documents. Alternately, we could normalize relevance scores to the range [0, 1], and then apply the same normalized score, such as 0.5, as a classification threshold across a variety of techniques and data sets.

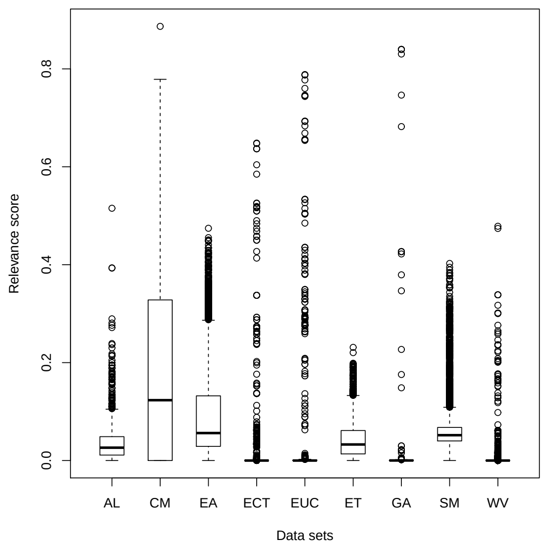


Fig. 3. Relevance scores: Global IDF approach

5.3 Discriminability

A metric that returns the same values for all compared techniques may not be a good metric, unless it is the case that multiple techniques actually do return the same degree of accuracy. It is clearly important to determine whether evaluation methods are sensitive enough to discriminate between high and low accuracy techniques. *Discriminability* is the ability to summarize the performance of a technique in order to distinguish between high accuracy and low accuracy techniques. It is similar to the non-coarseness property defined by Weyuker [34] for software complexity metrics.

Examples In Table 2, the bolded values indicate that the metric values for each data set are equal when rounded to the second decimal place. We can observe that precision at recall 1.0 was the same for many data sets, while other metrics provide different values in most cases.

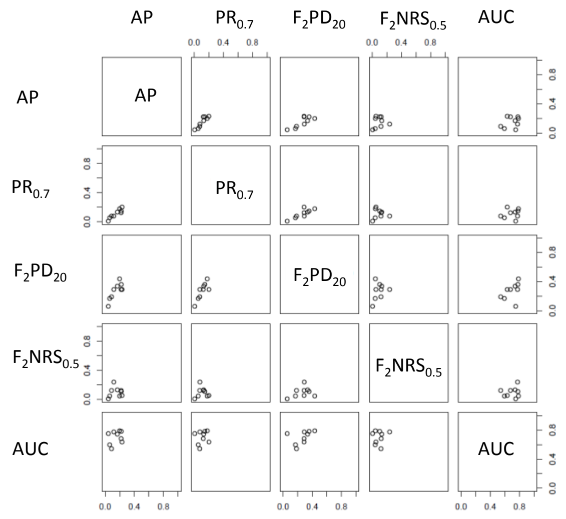


Fig. 4. Correlations between metrics: Global IDF approach

Table 3. Correlations between Metrics

	Global IDF			Local IDF			Rocchio		
	AP	**Lag**	**AUC**	**AP**	**Lag**	**AUC**	**AP**	**Lag**	**AUC**
AP	1	-0.58	0.23	1	**-0.97**	0.52	1	**-0.87**	0.57
Lag	-	1	**-0.75**	-	1	-0.40	-	1	-0.52
$\mathbf{PR_{0.1}}$	**0.75**	-0.63	0.48	**0.83**	**-0.85**	0.40	0.58	-0.33	0.03
$\mathbf{PR_{0.2}}$	**0.83**	**-0.77**	**0.67**	**0.98**	**-0.93**	0.48	**0.95**	**-0.75**	0.57
$\mathbf{PR_{0.3}}$	**0.93**	**-0.73**	0.42	**0.97**	**-0.90**	0.45	**0.97**	**-0.80**	0.62
$\mathbf{PR_{0.4}}$	**0.87**	**-0.77**	0.45	**0.98**	**-0.95**	0.50	**0.98**	**-0.83**	0.52
$\mathbf{PR_{0.5}}$	**0.87**	**-0.77**	0.45	**0.95**	**-0.90**	0.45	**0.98**	**-0.85**	0.55
$\mathbf{PR_{0.6}}$	**0.95**	**-0.67**	0.33	**0.95**	**-0.95**	0.50	**0.97**	**-0.87**	0.58
$\mathbf{PR_{0.7}}$	**0.92**	**-0.68**	0.35	**0.82**	**-0.82**	0.52	**0.77**	**-0.87**	0.55
$\mathbf{PR_{0.8}}$	**0.87**	-0.65	0.30	**0.85**	**-0.88**	0.28	**0.77**	**-0.87**	0.55
$\mathbf{PR_{0.9}}$	**0.87**	-0.50	0.12	**0.72**	**-0.77**	-0.17	0.60	**-0.75**	0.30
$\mathbf{PR_{1.0}}$	**0.82**	-0.27	0.17	0.3	-0.33	-0.27	0.35	-0.33	-0.08

Guidelines Benchmarking metrics should return values that differentiate between the accuracy of various techniques. When multiple metrics serve similar purposes, the metric that provides low discriminability should be avoided.

5.4 Orthogonality

Orthogonality is the ability of a set of evaluation methods to evaluate different aspects of a traceability technique. Simply utilizing a slew of evaluation metrics creates confusion in interpreting the results, and certainly undermines the goal of utilizing benchmark results to help practioners make informed decisions. For example, utilizing three different metrics at ten different threshold levels summarized in three different ways, will return 90 different results. When there are numerous ways to measure similar aspects (e.g. AP, Lag, and AUC to measure the satisfaction of Goal 3), it is helpful to examine the relationships between

Table 4. Comparison between Summarization Methods

	Average				**Aggregation**				**Wilcoxon Signed Rank Test**			
Data	**AP**	**Lag**	**AUC**	**PR$_{1.0}$**	**AP**	**Lag**	**AUC**	**PR$_{1.0}$**	**AP**	**Lag**	**AUC**	**PR$_{1.0}$**
AL	R[a]>L[b]>G[c]	R>L>G	R=L>G	R>L>G	R>L>G	R>L>G	R=L>G	R=L=G	RL LG RG	RL LG RG	RL LG RG	RL LG RG
CM	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R>L=G	RL **LG** **RG**	RL **LG** **RG**	RL **LG** **RG**	RL **LG** **RG**
EA	R>L>G	L>R>G	R>L>G	R=G>L	R>L>G	L>G>R	L>G>R	R=L=G	**RL** LG **RG**	RL **LG** RG	RL **LG** RG	**RL** **LG** RG
ECT	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R=L>G	**RL** LG RG	**RL** LG RG	**RL** LG RG	**RL** LG RG
EUC	R>L>G	R>L>G	R>L>G	R=L>G	R>L>G	R>L>G	R>L>G	R>L>G	**RL** **LG** **RG**	**RL** **LG** **RG**	**RL** **LG** **RG**	**RL** **LG** **RG**
ET	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R=L=G	**RL** **LG** **RG**	**RL** **LG** **RG**	**RL** **LG** **RG**	RL **LG** **RG**
GA	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R=L>G	**RL** LG **RG**	**RL** **LG** **RG**	**RL** **LG** **RG**	**RL** LG RG
SM	R>L>G	R>G>L	R>G>L	G>R=L	R>L>G	R>G>L	R>G>L	R=L>G	**RL** **LG** **RG**	**RL** LG RG	**RL** LG RG	**RL** **LG** RG
WV	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R>L>G	R=L>G	R=L=G	**RL** **LG** **RG**	**RL** **LG** **RG**	**RL** **LG** **RG**	**RL** **LG** **RG**

[a] Rocchio [b] Local IDF [c] Global IDF

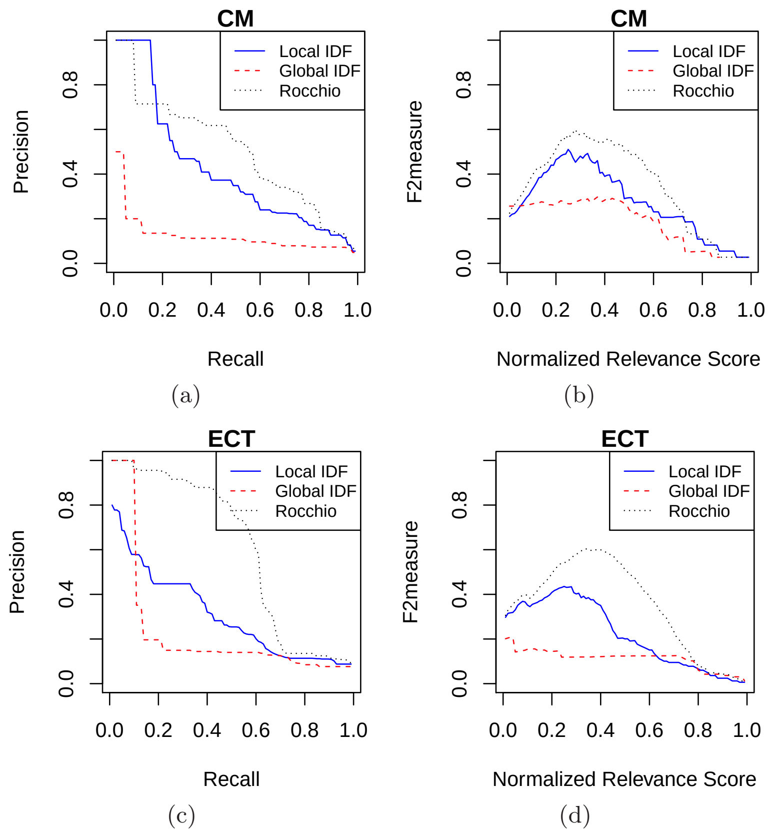


Fig. 5. Results from various threshold types

the different evaluation methods in terms of their similarities and differences, and then select the methods which highlight different benefits or aspects of the evaluated techniques.

Examples To evaluate orthogonality, we analyzed the correlations between various metrics computed from the tracing results of the Global IDF approach. The scatterplots in Figure 4 show the correlations between six metrics chosen based on the diversity of metrics and classification threshold types: average precision (AP), precision at recall 0.7 ($PR_{0.7}$), F_2-measure at 20% documents retreived (F_2PD_{20}), F_2-measure at normalized relevance score 0.5 ($F_2NRS_{0.5}$), and AUC. For example, the first row and the second column shows the correlation between AP and $PR_{0.7}$. Each point in each scatterplot represents a metric value computed from the retrieval results for one data set. The metric values were computed using the aggregation method.

In Figure 4, we can see that some metrics are highly correlated with others, implying that the tracing accuracy from the Global IDF method measured in the two correlated metrics are similarly high or low across data sets. For example, AP is strongly correlated with $PR_{0.7}$ and F_2PD_{20}, but is weakly correlated with $F_2NRS_{0.5}$ and AUC. Note that because the range of values of Lag was much greater than 1.0, we did not include Lag in Figure 4. Overall these results show that some of the metrics essentially measure very similar aspects.

Table 3 shows the Spearman rank correlation coefficients between AP, Lag, AUC, and precision measured at ten recall levels for the three traceability techniques. Correlations that were statistically significant at $p < 0.05$ are in bold face. In general, AP, Lag, and precision at most recall levels show high correlations except for AP and Lag of the Global IDF approach. AUC is weakly correlated with other metrics except for Lag for the Global IDF approach.

Note that there can be many alternate ways that orthogonality could have been measured. For example, we could have aggregated the results from multiple data sets and then compute the correlations between metrics from multiple techniques. However, examining all the possible ways to measure orthogonality is out of scope for this paper.

Guidelines When there are multiple candidate metrics, benchmarking results can be simplified but still informative when a small set of the metrics that highlight different aspects of the evaluated techniques are used. In our case study, AP, Lag, and precision measured at ten levels of recall were similar to each other and highly orthogonal to AUC in general.

5.5 Objectivity

Objectivity is the ability to evaluate a technique without being affected by subjective criteria. For example, it is hard to select and justify an objective and agreeable classification threshold for benchmarking, although it may be possible for a specific use of a technique in a specific organization.

Examples Figure 5 shows that arbitrary selection of thresholds leads to inconsistent conclusions using the tracing results from CM and ECT data sets. For example, in Figure 5(d), if an arbitrary decision had been made to measure accuracy at normalized relevance scores below 0.6, the Local IDF approach returns higher accuracy than the Global IDF approach, but on the other hand if the decision had been to measure accuracy at normalized relevance scores greater than 0.6, the techniques would appear to be approximately equal. Such inconsistency can be observed across all graphs in Figure 5.

In fact, a threshold should be selected according to the intended usage. For example, a project team that is willing to examine a large number of retrieved links could establish a low normalized relevance score or a high recall level as a threshold. Unless the context of actual usage can be objectively defined, the benchmarking result should not depend on such a threshold. The only broadly acceptable threshold might be to measure precision at perfect recall; however as discussed earlier, precision at perfect recall measured using the aggregation method suffers from low discriminability.

Table 5. Property Satisfaction Checklist

Metric	**Goal satisfiability**	**Summary Method**	**General.**	**Discriminability**	**Orthogonality**	**Object.**	**Robust.**
AP	Goal 3	Average	Yes	High	AUC>**Lag**>$\mathbf{PR}_{1.0}$	Yes	Low
		Aggregation	Yes	High	{AUC>Lag},$PR_{1.0}$	Yes	Low
		Per-query	Yes	Medium	N/A	Yes	High
Lag	Goal 3	Average	Yes	High	{$\mathbf{PR}_{1.0}$≥**AP**},AUC	Yes	Low
		Aggregation	Yes	High	$PR_{1.0}$>{AUC, AP}	Yes	Low
		Per-query	Yes	Medium	N/A	Yes	High
AUC	Goal 3	Average	Yes	Medium	AP, Lag, $PR_{1.0}$	Yes	Low
		Aggregation	Yes	Medium	$PR_{1.0}$>{AP, Lag}	Yes	Low
		Per-query	Yes	Medium	N/A	Yes	High
$\mathbf{PR}_{1.0}$	Goals 1,2	Average	Yes	Medium	AUC>**Lag**>**AP**	Yes	Low
		Aggregation	Yes	Low	AUC>{AP,Lag}	Yes	Low
		Per-query	Yes	Medium	N/A	Yes	High
$\mathbf{PR}_{<1.0}$	Goals 1,2	Average	Yes	Medium to High	N/A	No	Low
		Aggregation	Yes	Medium to High	N/A	No	Low
		Per-query	Yes	Medium	N/A	No	High
$\mathbf{F_2NRS}_{\leq 1.0}$	-	Average	Yes	Medium to High	N/A	No	Low
		Aggregation	Yes	Medium to High up to threshold 0.9 / Low at threshold 1.0	N/A	No	Low
		Per-query	Yes	Low to Medium up to threshold 0.5 / Low at threshold over 0.5	N/A	No	High
$\mathbf{F_2PD}_{\leq 100}$	-	Average	Yes	High to Medium up to threshold 50% / Low at threshold over 50%	N/A	No	Low
		Aggregation	Yes	Medium up to threshold 50% / Low at threshold over 50%	N/A	No	Low
		Per-query	Yes	Low	N/A	No	High
$\mathbf{F_2RS}_{\leq 1.0}$	-	Average	No	N/A[a]	N/A	No	Low
		Aggregation	No	N/A	N/A	No	Low
		Per-query	No	N/A	N/A	No	High

[a] Not Applicable

Guidelines Evaluation methods for benchmarking should not be affected by the context of use. A possible solution is to use a metric such as average precision which does not require a threshold.

5.6 Robustness

Robustness is the ability to measure the essential accuracy of a technique not affected by random chance due to peculiarities in a data set such as outliers.

Examples The results in the previous sections were all computed using the aggregation method. However, as explained in Section 3, traceability research has also used the average or per-query methods. Note that we explained other properties using results computed by the aggregation method simply because the method is easy to use and its use did not impact our discussion. With the average and aggregation methods, however, results can be affected by a small number of queries that return markedly better or worse results than other queries. To avoid this problem, we can either compute accuracy query-by-query and then visually inspect outliers using boxplots or we can use non-parametric statistical tests that are not affected by outliers and that make no assumption on the distribution of the data [5, 14].

Table 4 shows the tracing results measured in AP, Lag, AUC, and precision at recall 1.0 using the three summarization methods. For a statistical test from per-query results, we used the Wilcoxon signed rank test [14], a non-parametric alternative to the paired t-test, to compare each pair of techniques. In Table 4, A>B means that technique A traced more accurately than technique B for the average and aggregation methods. A=B indicates that techniques A and B provided the same accuracy when rounded to the second decimal place. For the per-query method, AB in bold indicates that techniques A and B provided statistically significantly different accuracy at $p < 0.05$.

From the results, we can observe that the ranks of techniques computed using the average and aggregation methods are coincident in general, but not always. In AP, Lag, and AUC, the Rocchio approach performed best and the Global IDF approach performed worst in many cases. Deviation from this tendency is presented in gray in Table 4. In all cases that the Global IDF approach performed better than other approaches or the Local IDF approach performed better than the Rocchio approach, the differences were not statistically significant for the three metrics. These results show the importance of employing a robust method such as statistical tests to derive a sound conclusion.

Guidelines Benchmarking should use statistical tests instead of comparing a single, summarized value.

5.7 Property Satisfaction Summary

Table 5 summarizes the extent to which each property is satisfied for each of the evaluation methods in our case study. Goal satisfiability, generalizability, and objectiveness were determined based on the definitions of the metrics and metric computation methods. Robustness was determined from both empirical results and common knowledge in statistics. Discriminability and orthogonality were examined empirically from our case study. Discriminability was measured relatively by counting the number of different metric values for the average and aggregation methods and the number of the statistical tests that exhibit significant differences for the per-query method using the Wilcoxon rank sum test. Among the 27 comparisons (nine data sets × three compared pairs of techniques), a metric that provided unique values or statistical differences for all comparisons was considered to have high discriminability. A metric that provided unequal values or statistical differences over 50% of the comparisions was considered to have medium discriminability. Otherwise, a metric was considered to have low discriminability. Note that the criteria of this classification is somewhat arbitrary, but the purpose of this classificaton is to determine the relative discriminability, not to compute precise discriminability. For orthogonality, A>B indicates that metric A is more orthogonal to the metric in the first column than metric B across all three of the techniques, and (A,B) indicates that no clear pattern in orthogonality between metrics A and B was observed. Metrics in bold face indicate that the least Spearman rank correlation coefficient between the bolded metric and the metric in the first column for the three techniques is over 0.6.

5.8 Applying the Properties

We now use the properties to select an appropriate evaluation method within the context of a specific benchmarking task.

Benchmarking Task The main goal of the planned benchmarking is to evaluate the accuracy of a newly proposed traceability technique for tracing safety requirements to test cases. Results will be compared to an existing state-of-the-art technique. Because the adverse impact of missing a safety requirement is high, possibly contributing to loss of lives, we need to ensure that all correct links are retrieved, even though this will increase the total number of returned links and increase the human evaluation effort. To minimize this effort, high precision is also important. Therefore, the ultimate goal is to select a traceability technique that provides high precision with close to perfect recall.

Selecting Evaluation Methods After investigating the existing metrics, we find that $PR_{1.0}$ meets our *goal satisfiability* property. However, from Table 5, we know that $PR_{1.0}$ measured with the average and aggregation methods exhibits low robustness, therefore we select $PR_{1.0}$ measured with the Wilcoxon rank sum test for further consideration (*robustness*). This approach provided medium level discriminability in our case study (*discriminability*). At the same time, because $PR_{1.0}$ has objectivity, there is no concern regarding its subjectivity (*objectivity*). $PR_{1.0}$ is also applicable across the compared techniques as well as all the data sets (*generalizability*). Because $PR_{1.0}$ is the only metric in Table 5 that satisfies the goal of this benchmarking, examining metrics orthogonal to $PR_{1.0}$ is not necessary. For additional analysis, AUC may be helpful for understanding a different aspect of the benchmarked technique (*orthogonality*). Finally, $PR_{1.0}$ measured using the Wilcoxon rank sum test was chosen as the evaluation method because it satisfies all the necessary properties. If the new technique provides statistically significantly better performance than the state-of-the-art technique, we adopt it for this tracing task. For informational purposes, we can report secondary evaluation results using other metrics.

Note that if a benchmarking task is evaluated with $PR_{1.0}$ using the aggregation method commonly adopted in current traceability research practices, the conclusions may be very different from the case when the evaluation is made with $PR_{1.0}$ using a statistical test. If the benchmarking was conducted using F-measure following the previous trend in traceability research, the benchmarking would measure neither Goals 1 nor 2, and would suffer from the need to subjectively select a threshold value. Our framework reduces the likelihood of evaluation problems by systematically and holistically guiding the assessment and selection of candidate evaluation methods using well-defined properties.

6 Threats to Validity

Given the difficulty of obtaining traceability data sets for use in the public domain, the data sets used in this case study are smaller than most industry data sets. Therefore, as the community builds a better repository of shared datasets, the datasets could be used to re-check the properties presented in this paper. Secondly, reaching consensus on an answer set is by no means trivial, as there is a class of links which are somewhat borderline in nature and could be classified either as correct or incorrect. However, this problem is at least partially mitigated by the fact that the answer sets used in this study are fairly stable because they have been used in multiple prior studies.

As stated earlier in the paper, our original intent is that the properties we describe be generalizable beyond traceability. However, the case study against which they were evaluated was taken from the traceability domain. To demonstrate applicability to a broader set of software engineering domains we need to apply and evaluate our properties in other domains. This is especially important because discriminability and orthogonality are evaluated empirically and can be affected by different techniques and datasets. Therefore, evaluation of discriminability and orthogonality is more about engineering based on the accumulated knowledge in the benchmarking domain rather than a precise science, and should be continuously updated. Nevertheless, our framework can be used as a vehicle to capture accumulated knowledge and to systematically guide the selection and creation of evaluation methods. Our focus in this paper is not on providing specific evaluation methods for the proposed properties, but on proposing the properties and explaining, using practical examples, the necessity of examining benchmarking evaluation methods according to those properties.

7 Summary

In this study, we performed a systematic literature review to investigate the current practices of software traceability technique evaluation and defined a framework that consists of six properties that benchmarking evaluation methods should possess.

This study was motivated by, and explained through, an example of benchmarking software traceability techniques. However, we believe the properties presented in this study can be broadly applied to other software engineering disciplines, but we do not claim that the identified properties are an exhaustive list. Future work includes improving the framework by examining evaluation methods for benchmarking techniques in other software engineering areas, obtaining feedback from the research community to collect accumulated wisdom for benchmarking and to find comprehensive properties, and suggesting possible methods for evaluating the satisfaction of the proposed properties.

8 Acknowledgments

The work described in this paper was funded in part by NSF under grant CCF-0811140.

References

1. Software Cost-effective Change and Evolution Research Lab Data Repository, http://web.soccerlab.polymtl.ca/ser/home.html.
2. Center of Excellence for Software Traceability, http://http://www.coest.org/, March 2008.
3. TEFSE challenge, The 6th Internation Workshp on Traceability in Emerging Forms of Software Engineering, http://www.cs.wm.edu/semeru/tefse2011/challenge.htm, 2011.
4. G. Antoniol, G. Canfora, G. Casazza, A. D. Lucia, and E. Merlo. Recovering traceability links between code and documentation. *IEEE Trans. on Software Eng.*, 28(10):970–983, 2002.
5. A. Arcuri and L. C. Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proc. of Intn'l Conf. on Software Eng.*, ICSE '11, pages 1–10, 2011.
6. R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley, 1999.
7. B. Berenbach, D. Gruseman, and J. Cleland-Huang. Application of just in time tracing to regulatory codes. In *Conference on Systems Engineering Research*, 2010.
8. L. Briand, S. Morasca, and V. Basili. Property-based software engineering measurement. *IEEE Trans. on Software Eng.*, 22(1):68–86, January 1996.
9. J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker. A machine learning approach for tracing regulatory codes to product specific requirements. In *Proc. of Intn'l Conf. on Software Eng.*, pages 155–164, 2010.
10. J. Cleland-Huang, A. Czauderna, A. Dekhtyar, O. Gotel, J. H. Hayes, et al. Grand challenges, benchmarks, and TraceLab: developing infrastructure for the software traceability community. In *Proc. of the 6th Intn'l Workshop on Traceability in Emerging Forms of Software Eng.*, pages 17–23. ACM, 2011.
11. J. Cleland-Huang, Y. Shin, E. Keenan, A. Czauderna, G. Leach, et al. Toward actionable, broadly accessible contests in software engineering. In *34th Intn'l Conf. on Software Eng., NIER, (to appear)*, 2012.
12. D. Cuddeback, A. Dekhtyar, and J. Hayes. Automated requirements traceability: The study of human analysts. In *Proc. of Requirements Eng. Conf.*, pages 231–240, Los Alamitos, CA, USA, 2010.
13. A. Czauderna, M. Gibiec, G. Leach, Y. Li, Y. Shin, et al. Traceability challenge 2011: using TraceLab to evaluate the impact of local versus global idf on trace retrieval. In *6th Intn'l Workshop on Traceability in Emerging Forms of Software Engineering*, pages 75–78, 2011.
14. J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30, 2006.

15. T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.
16. M. Gethers, R. Oliveto, D. Poshyvanyk, and A. D. Lucia. On integrating orthogonal information retrieval methods to improve traceability recovery. In *Proc. of Intn'l Conf. on Software Maintenance*, 2011.
17. O. Gotel and C. Finkelstein. An analysis of the requirements traceability problem. In *Proc. of Requirements Eng. Conf.*, pages 94 –101, Apr 1994.
18. J. H. Hayes, A. Dekhtyar, and S. K. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *IEEE Trans. on Software Eng.*, 32(1):4–19, 2006.
19. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22:5–53, January 2004.
20. E. A. Holbrook, J. Huffman Hayes, and A. Dekhtyar. Toward automating requirements satisfaction assessment. In *Proc. of Requirements Eng. Conf.*, pages 149–158, 2009.
21. E. Keenan, A. Czauderna, G. Leach, J. Cleland-Huang, Y. Shin, et al. TraceLab: An experimental workbench for equipping researchers to innovate, synthesize, and comparatively evaluate traceability solutions. In *Proc. of the 34th Intn'l Conf. on Software Eng., formal demo, (to appear)*, 2012.
22. B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. *Technical Report*, 2007.
23. B. Kitchenham, O. Pearlbrereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic literature reviews in software engineering a systematic literature review. *Information and Software Technology*, 51(1):7–15, 2009.
24. A. D. Lucia, R. Oliveto, and P. Sgueglia. Incremental approach and user feedbacks: a bullet for traceability recovery. In *Proc. of Intn'l Conf. on Software Maintenance*, pages 299–309, 2006.
25. Y. Ma and B. Cukic. Adequate and precise evaluation of quality models in software engineering studies. In *Proc. of the 3rd Intn'l Workshop on Predictor Models in Software Eng.*, Washington, DC, USA, 2007.
26. C. D. Manning, P. Raghavan, and H. Schuätze. *Introduction to Information Retrieval*. Cambridge University Press, NY, USA, 2008.
27. A. Marcus and J. Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. In *Proc. of Intn'l Conf. on Software Eng.*, pages 125 – 135, may 2003.
28. R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *Proc. of 18th IEEE Intn'l Conference on Program Comprehension (ICPC'10)*, pages 68–71, 2010.
29. B. Ramesh and M. Jarke. Toward reference models of requirements traceability. *IEEE Trans. Software Eng.*, 27(1):58–93, 2001.
30. T. Saracevic. Evaluation of evaluation in information retrieval. In *Proc. of the 18th annual intn'l ACM SIGIR conf. on Research and development in information retrieval*, pages 138–146, 1995.
31. Y. Shin and J. Cleland-Huang. A comparative evaluation of two user feedback techniques for requirements trace retrieval (to appear). In *Proceedings of the 27th Symposium on Applied Computing*, 2012.
32. S. E. Sim, S. Easterbrook, and R. C. Holt. Using benchmarking to advance research: a challenge to software engineering. In *Proc. of Intn'l Conf. on Software Eng.*, ICSE '03, pages 74–83, 2003.

33. S. K. Sundaram, J. H. Hayes, A. Dekhtyar, and E. A. Holbrook. Assessing traceability of software engineering artifacts. *Requir. Eng.*, 15:313–335, 2010.
34. E. Weyuker. Evaluating software complexity measures. *IEEE Trans. on Software Eng.*, 14(9):1357 –1365, 1988.
35. E. J. Weyuker, R. M. Bell, and T. J. Ostrand. We're finding most of the bugs, but what are we missing? In *Proc. of the 3rd Intn'l Conf. on Software Testing, Verification and Validation*, pages 313–322, 2010.